PATENT
Attorney Docket No. 23452.150
Client Reference No. LOT9-2001-0034-US1

SYSTEM AND METHOD FOR PRIORITY BASED APPLICATION SERVER UPDATES

Field of the Invention

This invention is a system and method for updating applications on a running application server without requiring a server restart.

Background of the Invention

As used herein, the term application is used to refer to a set of files that are useable in a processor based environment. In a network environment, an environment of a several processors in communication with one another, it is often difficult and inconvenient to update an application.

Typically, the process to update an application entails at least three steps: first, the network file server must be shut down, second, the application files are updated, and finally, the server must be restarted. One drawback this typical process is that for the period of time it takes to update the application, the server, and all other applications that reside on the server, are inaccessible to network clients.

Another drawback of existing application update mechanisms is that they often must be scheduled to occur at inconvenient times for network administrators. Because they entail a service interruption, most application updates are scheduled to take place at off peak hours (e.g., after midnight). This can be inconvenient and stressful for the personnel responsible for administering the updates.

Another drawback of existing application update mechanisms is that they require personnel to oversee the update. For example, an administrator must be present to shut down the server, ensure that the appropriate files are updated and restart the server.

Other drawbacks of current application update systems also exist.

Summary of the Invention

The present invention provides a system and method for updating applications without the need to interrupt server operation. The system may comprise an update deployer that functions as a main execution engine and is responsible for coordinating updates with an application server.

One advantage of the present invention is that it enables application updates with less server down time and a reduced amount of administrative work. According to some embodiments, a method for updating an application server may comprise configuring the deployer and enabling it to update the files.

These and other features and advantages of the invention will be apparent through the detailed description of the preferred embodiments and the drawings attached hereto. It is also to be understood that both the foregoing general description and the following detailed description are exemplary and not restrictive of the scope of the invention.

Brief Description of the Drawings

FIG. 1 is a schematic representation of the overall system 100 according to some embodiments of the invention.

FIG. 2 is a schematic flow diagram illustrating a method of updating an application according to some embodiments of the invention.

Detailed Description of the Drawing Figures

Figure 1 is a schematic representation of the overall system 100 according to some embodiments of the invention. The system may comprise a deployer 110 that functions as an execution agent as described herein. Deployer 110 may comprise a software module or other suitable processor readable agent that enables a processor to carry out the related functions.

Deployer 110 may communicate with other portions of system 100 via communication path 160. Communication path 160 may comprise any suitable signal carrying path. For example, communication path 160 may comprise a hard wired (e.g., cable) connection, a wireless (e.g., radio, satellite, cellular) connection, or some other suitable signal carrying path.

In some embodiments, Deployer 110 may communicate with a duplicate storage system or mirror 112. Mirror 112 may be a separate device (e.g., an additional server), in a separate location, or may comprise some other typical mirror server configuration.

System 100 may comprise at least one file server, such as application server 120. Application server 120 may comprise any suitable server or other mechanism for supplying files or services. For example, application server 120 may comprise a Java™ to platform enterprise edition (J2EE) web server, an IBM WebSphere™, BEA iPlanet™, IBM Domino™, or other suitable server.

Application server 120 may communicate (via a communication path 160) with appropriate system devices. For example, application server 120 may communicate with any number of storage devices 130a, 130n. Storage devices 130a, 130n may comprise databases or other file storage mechanisms. While storage devices 130a, 130n are shown in Figure 1 as separate devices, the invention is not so limited. Storage devices 130 a, 130n may comprise a portion of application server 120, a distributed network of storage devices, stand-alone storage devices, or some other storage arrangement.

Deployer 110, application server 120, mirror server 112, storage devices 130a, 130n and other system devices may communicate (e.g., via communication paths 160) directly with one another or over network 140. Network 140 may comprise any suitable network arrangement. For example, network 140 may comprise a local area network (LAN), a wide area network (WAN), a wireless network, an intranet, an extranet, the Internet, or some other suitable network configuration.

In addition, one or more client devices 150a, 150b, 150n may communicate over network 140. For example, client devices 150a, 150b, 150n may enable access to applications services by application server 120.

Figure 2 is a schematic flow diagram illustrating a method of updating an application according to some embodiments of the invention. The method shown in Figure 2 is but one example of the inventive method. Other arrangements of the steps listed in Figure 2 are possible.

At some time prior to initiating an update to an application, an administrator configures deployer 110 as shown in Figure 2 at 200. Configuring deployer 110 may comprise any suitable method for setting predetermined criteria dictating when and how application server 120 should handle updates.

Any suitable criteria may be used to configure deployer 110. For example, criteria relating to the polling rate at which deployer 110 checks for updates, priority levels for the updates, time of day to deploy updates, changes in priority level based upon time of day, changes in priority level based upon type of application, and other criteria may be configured at step 200.

Configuration step 200 may also comprise setting a priority for which updates may occur. Priority may be established with any suitable level of granularity. For example, priority may be divided into high, medium and low priority levels. In addition, priority may be set according to application, time of day, number of users, number of active sessions, type of application, type of update (e.g., minimal or significant change), combinations of one or more of the foregoing, or other predetermined criteria.

In some embodiments, the configuration step 200 for deployer 110 may persist as long as desired. In this manner, an administrator need only perform configuration step 200 once if desired. Of course, configuration step 200 may also occur as often as desired.

When an update to an application is to occur, a developer or an administrator may make the modifications to the relevant application files as indicated at 202. Modification may occur in any suitable manner. For example, an application may be modified by modifying the application files in mirror server 112 or in application server 120 storage space (e.g., storage device 130a, 130n).

Deployer 110 may be configured to automatically notice that an update is available. Deployer 110 may notice that updates are available using any suitable method. For example, deployer 110 may periodically poll the applications in mirror 112 or storage devices 130a, 103n, to determine if file modification times have changed. Other schemes may also be used to determine when updates have occurred.

If deployer 110 configuration is so set, an update may initiate at 206 when deployer 110 sends a message to application server 120 to request an update for the modified application. The message to application server 120 may also comprise an indication of the priority level for the update.

Application server 120 examines the message and determines an appropriate response as indicated at 208. The response is a yes or no determination of whether to proceed with the update as indicated at 210 and may be based upon the priority set in deployer 110 configuration files.

For embodiments where application server 120 response is based upon update priority, any suitable predetermined response scheme may be implemented. For example, application server 120 may respond to a high priority message with a yes-proceed answer.

Medium priority update messages may receive a yes-proceed answer when certain criteria are satisfied. For example, a medium priority update may proceed when no users (e.g., clients 150a, 150b, 150n) have active sessions in the application. Other measures may also be implemented to insure that medium priority updates eventually proceed. For example, once a medium priority update request is received, application server 120 may refuse to accept new user sessions for the application; active users may continue to use the application, but new users are not allowed until the update is completed. Similarly, application server 120 may respond with a yes-proceed answer to low priority update messages when no active user sessions are present. Other application server 120 response schemes may also be used.

Application server 120 response may be determined as indicated at 210. If the determination at 210 is not to proceed, deployer 110 may cancel the update as indicated at 212. Deployer 110 may remember the failure to proceed and periodically retry the update request, as indicated in Figure 2, or may implement some other procedure as dictated by deployer 110 configuration.

If the determination at 210 is to proceed, application the application or applications which are to be updated are adjusted as indicated at 214. Adjustment of the application or applications which are to be updated may occur in any suitable fashion as dictated by deployer 110 configuration. For example, as discussed above, proceeding with a high priority updates may cause application server 120 to halt the application; proceeding with medium priority updates may cause application server 120 to prevent new user sessions and halt the applications when the in-use sessions terminate; proceeding with low priority updates may cause application server to wait until no active sessions are present and then halt the applications. Other adjustments are possible.

The application update is implemented at 216. Implementation of the update may be accomplished in any suitable manner. For example, updated files on mirror server 112 or in application storage space (e.g., storage devices 130a, 130n) may be copied onto application server's 120 applications store. Other methods of implementing the update may also be used.

Deployer 110 may signal application server 120 to indicate that the update is complete as indicated at 218. Application server 120 may then restart the updated application as indicated at 220.

Other embodiments, uses and advantages of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. The specification should be considered exemplary only, and the scope of the invention is accordingly intended to be limited only by the following claims.